SetGNER: General Named Entity Recognition as Entity Set Generation

Yuxin He[1] and **Buzhou Tang**[1,2,*]
[1]Department of Computer Science, Harbin Institute of Technology, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China
`21S051047@stu.hit.edu.cn`
`tangbuzhou@gmail.com`

Abstract

Recently, joint recognition of flat, nested and discontinuous entities has received increasing attention. Motivated by the observation that the target output of NER is essentially a set of sequences, we propose a novel entity set generation framework for general NER scenes in this paper. Different from sequence-to-sequence NER methods, our method does not force the entities to be generated in a predefined order and can get rid of the problem of error propagation and inefficient decoding. Distinguished from the set-prediction NER framework, our method treats each entity as a sequence and is capable of recognizing discontinuous mentions. Given an input sentence, the model first encodes the sentence in word-level and detects potential entity mentions based on the encoder's output, then reconstructs entity mentions from the detected entity heads in parallel. To let the encoder of our model capture better right-to-left semantic structure, we also propose an auxiliary Inverse Generation Training task. Extensive experiments show that our model (w/o. Inverse Generation Training) outperforms state-of-the-art generative NER models by a large margin on two discontinuous NER datasets, two nested NER datasets and one flat NER dataset. Besides, the auxiliary Inverse Generation Training task is found to further improve the model's performance on the five datasets.

1 Introduction

Named entity recognition (NER) is a fundamental task in the field of information extraction and has played an important role in the development of natural language processing. There exist three well-studied subtasks of NER, i.e. flat NER, nested NER and discontinuous NER, as illustrated in Figure 1.

Recently, researchers have grown more interest in tackling the three subtasks jointly, which we refer to as general NER (Li et al., 2021; Dai et al., 2020; Yan et al., 2021). Existing frameworks for general NER fall into three categories: (1) span-based models; (2) sequence-to-sequence (seq2seq) models; (3) models based on other techniques like hyper-graphs and shift-reduce parsers. Among them, seq2seq models (Fei et al., 2021; Yan et al., 2021) have demonstrated SOTA performance. However, they organize target entities into a single sequence according to a predetermined order. This formulation violates the fact that the target entities are inherently a unordered set, and introduces an incorrect bias (entity-order confounder) to the model (Tan et al., 2021; Zhang et al., 2022). In addition, the conduct of generating target entities sequentially suffers from two negative side effects: (1) Low inference speed; (2) Error propagation, i.e. previous errors will result in a misleading context for current generation step.

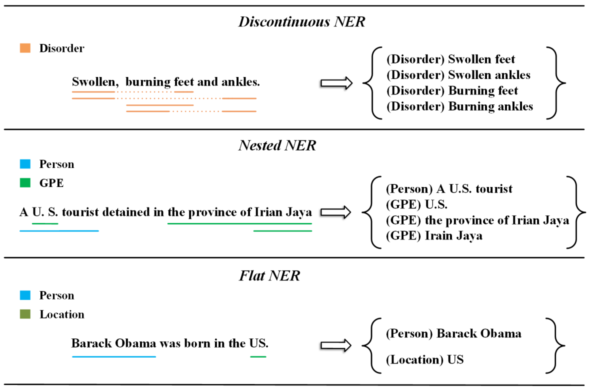

Discontinuous NER
Disorder
Swollen, burning feet and ankles.
(Disorder) Swollen feet
(Disorder) Swollen ankles
(Disorder) Burning feet
(Disorder) Burning ankles
Nested NER
Person
GPE
A U. S. tourist detained in the province of Irian Jaya
(Person) A U.S. tourist
(GPE) U.S.
(GPE) the province of Irian Jaya
(GPE) Irain Jaya
Flat NER
Person
Location
Barack Obama was born in the US.
(Person) Barack Obama
(Location) US

Figure 1: Examples of the discontinuous / nested / flat NER. It can be observed that entities may overlap with each other in general NER scenes and these overlapped entities are inherently unordered. Hence, instead of generating entities sequentially in a predefined order, it is more suitable to generate the set of entities concurrently.

In this paper, we abandon the linear design of target entity sequence adopted by previous generative NER methods, and come up with a novel Entity Set Generation framework, *SetGNER*. Given an input sentence, the framework first encodes the sentence

[*]Corresponding Author.

and detects potential entity mentions based on the encoder's output, then utilizes detected entity heads as a set of initial sequences to reconstruct the set of target entities. Note that, this procedure is similar to the way human beings perform NER. While reading, a human attends to potential mentions of entities subconsciously. When encountering a plausible start of entities, the mind will activate a cluster of neurons for it (Kemmerer, 2015), which consistently collects words related to it until all entities starting with it are recollected. These clusters of neurons just function as a distributed decoder that generates a set of entities in parallel.

The distributed nature of our decoding scheme also brings remarkable inference speed-up. With a decoding time complexity of $O(NL)$, where N is the sentence length and L is the average entity length, our model is efficient enough for both offline and online applications.

To correctly recognize the boundary of an entity, right-to-left semantic structure is as important as left-to-right semantic structure, which is ignored by previous generative NER research. Inspired by this, we additionally propose an auxiliary learning task — *Inverse Generation Training*. Guided by the inverse generation loss, the encoder can get more familiar with the right-to-left semantic structure of entity mentions, making it easier for the decoder to generate complete entity mentions.

Note that, our work is different from previous work proposed to address the problems faced with generative NER models. In Tan et al. (2021), a sequence-to-set network is proposed to predict the set of target entities in a sentence. However, it assumes that each entity is a span and cannot handle the recognition of discontinuous mentions. In contrast, our model treats each entity as a sequence and can naturally tackle the discontinuous mentions. Zhang et al. (2022) analyze two kinds of incorrect bias in the seq2seq NER models, i.e. pre-context confounder (when generating a word of an entity, a model can be affected by pre-generated words that has no causal relation with the word to be generated), entity-order confounder, and propose two data augmentation methods to address them. However, their model is still trained to generate entities sequentially after all.

To sum up, our main contributions include:

- We propose *SetGNER*, a novel Entity Set Generation framework for general NER scenes. Distinct from seq2seq models, it effectively gets rid of the entity-order confounder and error propagation caused by linearization assumption, and brings high inference speed-up.
- We also come up with a novel auxiliary learning task — *Inverse Generation Training*, to help the encoder of our model capture better right-to-left semantic structure.
- Experiment results show that our model outperforms SOTA seq2seq NER models by a large margin on two discontinuous NER datasets, two nested NER datasets and one flat NER dataset, while being 3 times faster than SOTA seq2seq NER models.

2 Problem Formulation

We uniformly formulate the task of recognizing flat / nested / discontinuous entities as a pointer-based Entity Set Generation problem.

A pointer is used for copying a source word or a class tag or a special tag into target sequence. There are three special tags: $\langle\varnothing\rangle$, indicating no-entity-found; $\langle\bigcirc\rangle$, indicating fragment of entity is found; $\langle$/s$\rangle$, indicating the end of the generated sequence. Suppose the number of entity classes is C. We first define the pointer to class tag $T_i (0 \le i < C)$ as Ptr$(T_i) = i$, the pointers to special tags $\langle\varnothing\rangle$, $\langle\bigcirc\rangle$, $\langle$/s$\rangle$ as Ptr$(\langle\varnothing\rangle) = C$, Ptr$(\langle\bigcirc\rangle) = C + 1$, Ptr$(\langle$/s$\rangle) = C + 2$.

Given a sentence with N words $[w_0, ..., w_{N-1}]$, the pointer to word w_j is then defined as:

$$\text{Ptr}(w_j) = C + 3 + j$$

To simplify annotation, we generalize the Ptr$(\cdot)$ operation to sequences of words / tags, i.e. Ptr$(\,[\cdot]\,) = [\,\text{Ptr}(\cdot)\,]$.

The target output is the set of entity pointer sequences $\left\{\text{Ptr}(e^i)\right\}_{i=1}^{M}$, where M is the number of entities in the input sentence, e^i is the i-th entity consisting of l words $w_{e^i_0}, w_{e^i_1}, ..., w_{e^i_{l-1}}$. And the pointer sequence for entity e^i is defined as:

$$\text{Ptr}(e^i) = \text{Ptr}([w_{e^i_0}, w_{e^i_1}, ..., w_{e^i_{l-1}}, T(e^i), \langle/\text{s}\rangle])$$

where $T(e^i)$ is the class tag of e^i, $\langle$/s$\rangle$ indicates the end of generated sequence.

Under this definition, each head word of entity is associated with a target sequence or multiple target sequences (when there are multiple entities starting with the head word). We additionally appoint a target sequence Ptr$([w_j, \langle\varnothing\rangle, \langle/\text{s}\rangle])$ to word w_j, if w_j

does not belong to any entity; or Ptr($[w_j, \langle\bigcirc\rangle, \langle/\text{s}\rangle]$) if w_j is not an head word but a fragment of entity.

3 Method

As shown in Figure 2, the proposed model consists of a word-level encoder, a mention detector and a parallel generator based on copying mechanism. We train it with a combination of generation loss, occurrence detection loss and entity part classification loss, and leverage an adaptive beam-search scheme during inference.

3.1 Word-level Encoder

The input sentence $[w_0, w_1, ..., w_{N-1}]$ is first tokenized into $[\langle\text{s}\rangle, t_1, t_2, ..., t_{N'}, \langle/\text{s}\rangle]$ using BPE (Sennrich et al., 2016) tokenizer, where N' is the length of content tokens. We denote the tokenized sentence as $\mathbf{X}$.

We then calculate the contextual representation of the tokenized sentence using pre-trained BART (Lewis et al., 2020) encoder:

$$\mathbf{H} = \text{Encoder}(\mathbf{X}) \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{|\mathbf{X}| \times d}$ and d is the dimension of our model.

To obtain the word-level representation of each word, we max-pool the contextual representations of its first and last tokens:

$$\mathbf{r}_i = \text{MaxPool}(\mathbf{H}[start_{w_i}], \mathbf{H}[end_{w_i}]) \tag{2}$$

$$\mathbf{R} = [\mathbf{r}_i]_{i=0}^{N-1} \tag{3}$$

where $\mathbf{r}_i$ is the representation of w_i, $start_{w_i}$ and end_{w_i} are the indexes of the first and last tokens of w_i respectively, $\mathbf{R} \in \mathbb{R}^{N \times d}$ is the word-level representation matrix.

3.2 Mention Detector

Occurrence Detection

Since there may be multiple entities sharing the same head word, we have to predict how many entities have occurred with the head word. This is achieved by conducting classification over each word, where the label of a word is the number of entities starting with it.

Concretely, we first transforms the word-level representation matrix into feature matrix $\mathbf{V}$:

$$\mathbf{V} = \text{ReLU}(\mathbf{W}_V \mathbf{R} + \mathbf{b}_V) \tag{4}$$

Then utilize a softmax layer to predict the number of occurrences:

$$\mathbf{P}^o = \text{softmax}(\mathbf{W}_o \mathbf{V} + \mathbf{b}_o) \tag{5}$$

The loss function for occurrence detection is defined as follows.

$$\begin{aligned}\mathcal{L}_o = - \sum_{i=0}^{N-1} \sum_{j=1}^{O_{max}} & \mathbb{1}(y_i^o = j)\log(\mathbf{P}_{ij}^o) \\ & + \mathbb{1}(y_i^o \neq j)\log(1 - \mathbf{P}_{ij}^o)\end{aligned} \tag{6}$$

where O_{max} is the maximum number of entities starting with the same head word in the dataset. Note that, casting the occurrence prediction problem as a regression task may be a better choice and we leave it for future research.

Entity Part Classification

Our model also detects potential parts of entities by predicting whether a word is the head/tail of any entity or not, and whether a word belongs to any entity or not. This is essentially a multi-label classification task, which can be handled by three binary classifiers as follows.

$$\mathbf{p}^h = \text{sigmoid}(\mathbf{W}_h \mathbf{V} + \mathbf{b}_h) \tag{7}$$

$$\mathbf{p}^t = \text{sigmoid}(\mathbf{W}_t \mathbf{V} + \mathbf{b}_t) \tag{8}$$

$$\mathbf{p}^b = \text{sigmoid}(\mathbf{W}_p \mathbf{V} + \mathbf{b}_b) \tag{9}$$

where h, t and b stand for "entity head", "entity tail" and "belonging to entity" respectively.

We utilize binary cross-entropy loss to optimize this module:

$$\begin{aligned}\mathcal{L}_e = - \sum_{\delta \in \{h,t,b\}} \sum_{i=0}^{N-1} & y_i^\delta \log(\mathbf{p}_i^\delta) \\ & + (1 - y_i^\delta)\log(1 - \mathbf{p}_i^\delta)\end{aligned} \tag{10}$$

3.3 Parallel Generation

The backbone of our decoder is the one proposed by Yan et al. (2021), which is based on BART decoder and equipped with copying mechanism.

Boundary-guided Initialization

During inference, the decoder first initiates the set of target sequences with pointers to detected entity heads. Concretely, the target sequence corresponding to entity head w_i is initialized as:

$$\hat{\mathbf{y}}^{(i)} := [\text{Ptr}(w_i)]\,,\ i \in \Omega_{head} \tag{11}$$

$$\Omega_{head} = \{i \mid \mathbf{p}_i^h \geq 0.5\} \tag{12}$$

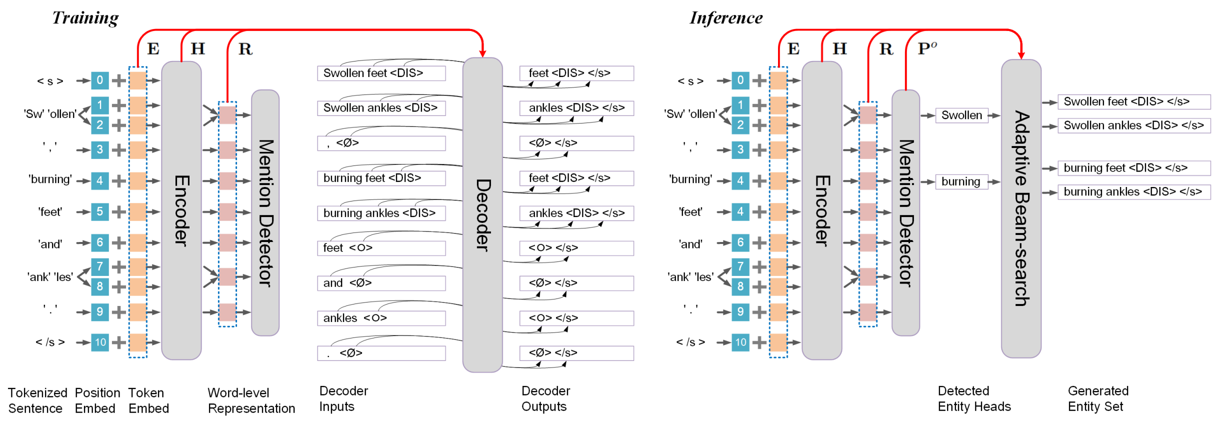


Figure 2: An overview of the proposed Entity Set Generation framework.

where Ω_{head} is the indexes of detected entity heads.

Such initialization provides explicit left boundaries for parallel generation, avoiding the negative influence of generation errors that may occur ahead of generating current entities.

Adaptive Position Embeddings

To guide our decoder with the position information of the sentence's region that each target sequence should focus on, we devise adaptive position embeddings for the decoder. Concretely, the position embedding for the t-th pointer in $\hat{\mathbf{y}}^{(i)}$ is defined as

$$\text{PosEmb}(\hat{\mathbf{y}}_t^{(i)}) = \text{DecoderPosEmb}[i \cdot \tau + t] \quad (13)$$

where τ is a hyper-parameter controlling the position interval, which intuitively represents the average number of tokens that each word holds.

Decoding

Since the target sequences are made up of pointers, they should be converted into words/tags before decoding. We denote this operation as Retrieve($\cdot$), which is defined as:

$$\text{Retrieve}(\hat{y}) = \begin{cases} T_{\hat{y}}, & \hat{y} < C \\ \langle\varnothing\rangle/\langle\bigcirc\rangle/\langle\text{/s}\rangle, & C \leq \hat{y} \leq C+2 \\ w_{\hat{y}-C-3}, & \hat{y} \geq C+3 \end{cases}$$

The decoder then calculates the hidden state of current time step for each target sequence via cross-attention over encoder output and self-attention over hidden states of previous time steps.

$$\mathbf{d}_t^{(i)} = \text{Decoder}(\mathbf{H}; \text{Retrieve}(\hat{\mathbf{y}}_{<t}^{(i)})) \quad (14)$$

The probability distribution of $\hat{y}_t^{(i)}$ is calculate via copying mechanism as follows:

$$P(\hat{y}_t^{(i)}) = \text{softmax}(\left[\mathbf{T}^E; \text{Emb}(\langle\varnothing\rangle, \langle\bigcirc\rangle, \langle\text{/s}\rangle); \widetilde{\mathbf{R}}\right] \odot \mathbf{d}_t^{(i)})$$
$$\mathbf{T}^E = [\text{Emb}(T_i)]_{i=0}^{C-1}$$
$$\widetilde{\mathbf{R}} = (\mathbf{R} + \mathbf{E})/2$$
$$\mathbf{E} = \left[\text{MaxPool}(\text{Emb}(\mathbf{X}_{start_{w_i}}), \text{Emb}(\mathbf{X}_{end_{w_i}}))\right]_{i=0}^{N-1}$$

where $\mathbf{T}^E$ is the learnable embeddings of class tags, $\widetilde{\mathbf{R}}$ is the combination of word-level representations $\mathbf{R}$ and word-level embeddings $\mathbf{E}$ (transformed from BART token embeddings).

We train the decoder in a teacher-forcing manner using all ground-truth target sequences defined in Section 2. The generation loss is as follows:

$$\mathcal{L}_g = -\sum_{i=0}^{N-1} \log P(\mathbf{y}^{(i)} | \mathbf{X}; \theta) \quad (15)$$

Note that, there can be multiple ground-truth sequences associated with w_i, in which case their losses are all summed up.

Adaptive Beam-search

Since there may exist multiple entities starting with the same head word, we devise an adaptive beam-search mechanism to generate multiple entity sequences sharing the same start. Concretely, we first utilize standard beam-search to generate a fixed number of candidate sequences for each detected head word w_i and then select the top-K_i candidate sequences as the entity sequences generated from w_i. K_i here is the predicted number of entity occurrences corresponding to w_i, i.e.,

$$K_i = \arg\max \mathbf{P}_i^o \quad (16)$$

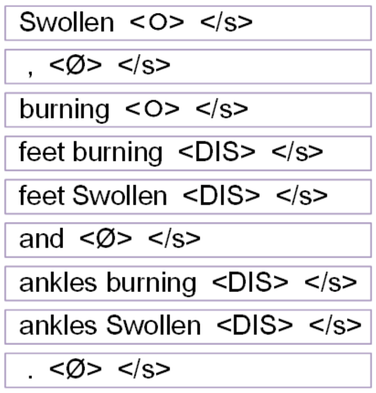


Figure 3: The set of inverse target sequences $\tilde{\mathbf{y}}$ for the instance "Swollen, burning feet and ankles."

3.4 Training

Inverse Generation Training
Since the forward generation task is biased towards the left-to-right semantic structure, we propose to help our model learn better right-to-left semantic structure via inverse generation training. To realize this, we instantiate an auxiliary decoder and train it to generate from entity tail to entity head. For the instance "Swollen, burning feet and ankles." , its inverse target sequences are defined as in Figure 3. And the inverse generation loss is calculated over all inverse target sequences $\tilde{\mathbf{y}}$:

$$\mathcal{L}_{\tilde{g}} = -\sum_{i=0}^{N-1} \log P(\tilde{\mathbf{y}}^{(i)}|\mathbf{X};\theta) \tag{17}$$

Note that, the auxiliary decoder is discarded after training phase.

Joint Learning
In each optimization step, we alternately train our model with the forward generation loss and inverse generation loss, together with the occurrence detection loss and entity part classification loss:

$$\mathcal{L} = \mathcal{L}_{g/\tilde{g}} + \mathcal{L}_o + \mathcal{L}_e \tag{18}$$

where $\mathcal{L}_{g/\tilde{g}}$ means alternating between forward generation loss and inverse generation loss.

4 Experiments

4.1 Datasets and Evaluation Details

We experiment on two discontinuous NER datasets CADEC and ShARe13, two nested NER datasets ACE04 and ACE05, and one flat NER dataset CoNLL03. Please refer to Appendix A for details and statistics of the five datasets.

For CADEC and ShARe13, we follow the same data split used in Dai et al. (2020); Yan et al. (2021). For ACE04 and ACE05, we use the same data split as in Muis and Lu (2017); Yu et al. (2020). For CoNLL03, we follow Yu et al. (2020); Yan et al. (2021) to concatenate the train and development sets. Strict evaluation metrics is applied, where an entity is confirmed correct only if its boundary and type label are both recognized correctly. Precision (P), Recall (R) and Micro F1 score (F1) are reported in the results. We report the average performance on 3 random seeds.

4.2 Implementation Details

We initialize model parameters from pre-trained BART-Large, which consists of 12 transformer blocks for encoder and 12 transformer blocks for decoder. The dimension size of the model is 1024. We use AdamW optimizer with different learning rates for encoder and decoder. Linear learning rate scheduling is employed. We fix the maximum number of words rather than the number of sentences in each batch, since the memory occupied by our model is determined by the number of words in a batch. Details of hyper-parameter tuning and settings are included in Appendix B.

4.3 Compared Methods

We compare our model principally with SOTA generative NER models and the set prediction NER model. See Section 5 for an introduction of them. Performances of SOTA discriminative NER models on the five datasets are also listed for reference. See Appendix D for an introduction of them.

4.4 Main Results

Discontinuous NER Table 1 shows the overall results of SetGNER's performance on discontinuous NER datasets. SetGNER outperforms the SOTA generative model (Fei et al., 2021) by +0.75 F1 and +0.33 F1 on CADEC and ShARe13 respectively, demonstrating the superiority of SetGNER on the task of discontinuous NER. After introducing Inverse Generation Training into our framework, the F1 scores of SetGNER further increase by +0.39 and +0.28.

Nested NER The results on nested NER tasks are shown in Table 2. SetGNER yields +0.26 and +0.45 F1 gains over the SOTA generative baseline (Lu et al., 2022) on ACE04 and ACE05. Compared with the SOTA set prediction model, SetGNER per-

Paradigm	Model	CADEC			ShARe13		
		P	R	F1	P	R	F1
Discriminative	Tang et al. (2018)	*67.8*	*64.9*	*66.3*	-	-	-
	Dai et al. (2020) [ELMO]	*68.9*	*69.0*	*69.0*	*80.5*	*75.0*	*77.7*
	Wang et al. (2021b)* [BERT-Large]	*70.5*	*72.5*	*71.5*	*83.3*	*77.0*	*80.2*
Generative	Yan et al. (2021) [BART-Large]	70.08	71.21	70.64	82.09	77.42	79.69
	Fei et al. (2021) [BERT-Large]	73.50	71.80	72.40	**84.90**	77.20	80.30
	Zhang et al. (2022)* [T5-Base]	71.35	71.86	71.60	81.09	78.13	79.58
	SetGNER [BART-Large]	**74.57**	71.78	73.15	83.58	77.82	80.63
	+ Inverse Generation Training	74.42	**72.72**	**73.56**	83.45	**78.48**	**80.91**

Table 1: Results on discontinuous NER datasets CADEC and ShARe13. $\star$ For a fair comparison on ShARe13, we replace the ClinicalBERT used by Wang et al. (2021b) with vanilla BERT-Large and rerun their code. $*$ Note that T5-Base has the same number of Transformer layers as BART-Large and is of a comparable size with BART-Large. The italic font indicates the results are only listed for reference. The bold font indicates the best score and the underline font indicates the second-best score.

Paradigm	Model	ACE04			ACE05		
		P	R	F1	P	R	F1
Descriminative	Yu et al. (2020)‡ [BERT-Large]	*87.30*	*86.00*	*86.70*	*85.20*	*85.60*	*85.40*
	Li et al. (2020) [BERT-Large]	*85.83*	*85.77*	*85.80*	*85.01*	*84.13*	*84.57*
	Xu et al. (2021) [BERT-Large]	*86.90*	*85.80*	*86.30*	*85.70*	*85.20*	*85.40*
	Shen et al. (2021)$^{\dagger\ \ddagger}$ [BERT-Large]	*87.44*	*87.38*	*87.41*	*86.09*	*87.27*	*86.67*
Set Prediction	Tan et al. (2021)$^{\dagger\ \ddagger}$ [BERT-Large]	*88.46*	*86.10*	*87.26*	*87.48*	*86.63*	*87.05*
	Tan et al. (2021)* [BERT-Large]	**88.05**	85.78	86.94	**86.73**	86.08	86.38
Generative	Straková et al. (2019) [BERT-Large]	-	-	84.40	-	-	84.33
	Yan et al. (2021) [BART-Large]	87.27	86.41	86.84	83.16	86.38	84.74
	Zhang et al. (2022) [T5-Base]	86.36	84.54	85.44	82.92	**87.05**	84.93
	Lu et al. (2022) [UIE (T5-Large)]	-	-	86.89	-	-	85.78
	SetGNER [BART-Large]	87.61	86.69	87.15	85.88	86.40	86.23
	+ Inverse Generation Training	87.45	**87.14**	**87.37**	85.86	86.92	**86.50**

Table 2: Results on nested NER datasets ACE04 and ACE05. $\dagger$ means leveraging extra embeddings (e.g. GloVE, character embeddings and POS embeddings). $\ddagger$ means leveraging extra context. $\star$ means we rerun their code without using extra embeddings or extra context.

Model	CoNLL03		
	P	R	F1
Akbik et al. (2019)† [BERT-Large]	-	-	*92.8*
Li et al. (2020) [BERT-Large]	*92.3*	*94.6*	*93.0*
Shen et al. (2021)$^{\dagger\ \ddagger}$ [BERT-Large]	*92.1*	*93.7*	*92.9*
Wang et al. (2021a)‡ [BERT-Large]	-	-	*93.2*
Tan et al. (2021)* [BERT-Large]	92.3	93.8	93.0
Straková et al. (2019)† [BERT-Large]	-	-	93.1
Yan et al. (2021) [BART-Large]	92.6	**93.9**	**93.2**
Zhang et al. (2022) [T5-Base]	**92.8**	93.5	93.1
Lu et al. (2022) [UIE (T5-Large)]	-	-	93.0
SetGNER [BART-Large]	92.7	93.4	93.1
+ Inverse Generation Training	**92.8**	93.6	**93.2**

Table 3: Results on CoNLL03. $\dagger$ means using extra embeddings. $\ddagger$ means using extra context. $\star$ means we run their code without using extra embeddings /context.

forms better on ACE04 (+ 0.21 F1) while performs competitively on ACE05. This demonstrates the effectiveness of SetGNER on tackling nested entities. Besides, Inverse Generation Training also boosts the model's performance on the two datasets.

Flat NER Table 3 shows the results of our model's performance on the CoNLL03 dataset. We can see that SetGNER is competitive with SOTA models on CoNLL03, verifying that our model can identify flat entities with high precision and high recall. When leveraging Inverse Generation Training, our model achieves SOTA performance on this NER benchmark.

4.5 Ablation Study

We conduct ablation study on CADEC and ACE04 to verify the effectiveness of different components

Model	CADEC			ACE04		
	P	R	F1	P	R	F1
Yan et al. (2021) [BART-Large]	70.08	71.21	70.64	87.27	86.41	86.84
Zhang et al. (2022) [T5-Base]	71.35	71.86	71.60	86.36	84.54	85.44
SetGNER [BART-Large]	74.57	71.78	73.15	87.61	86.69	87.15
w/o Word-level Representation	74.62	71.66	72.59 (-0.56)	87.21	86.60	86.91 (-0.24)
w/o Adapted Position Embedding	68.27	66.07	67.18 (-5.97)	76.38	74.09	75.37 (-11.8)
w/o Entity Part Classification	73.66	71.72	72.75 (-0.40)	87.18	86.53	86.94 (-0.21)
w/o Occurrence Detection	74.70	63.36	69.09 (-4.06)	87.79	70.53	73.46 (-13.7)

Table 4: Results of ablation study on CADEC and ACE04.

of SetGNER. The results are shown in Table 4. After replacing the **word-level representations** with the token-level representations and using token-level pointers, the F1 scores on the two datasets drop by 0.56 and 0.24 respectively, verifying the advantage of word-level representations and word-level pointers. After replacing the **adaptive position embeddings** with vanilla position embeddings, our model can not function properly. This demonstrates the necessity of guiding our decoder with the position information of the sentence's region that each target sequence should focus on. After removing the **entity part classification** module and generating from all words rather than detected head words, the F1-score of SetGNER drops by 0.40 and 0.21 on the two datasets, which means it is helpful to detect different parts of entities and generate selectively. Without the **occurrence detection** module (the adaptive beam-search degrades to standard beam-search), SetGNER can only generate one entity from each detected head word, and the recall of SetGNER drops significantly on the two datasets.

4.6 Analysis

We conduct a series of experiments to analyze the advantage of our Entity Set Generation model over seq2seq NER models and the benefit of Inverse Generation Training.

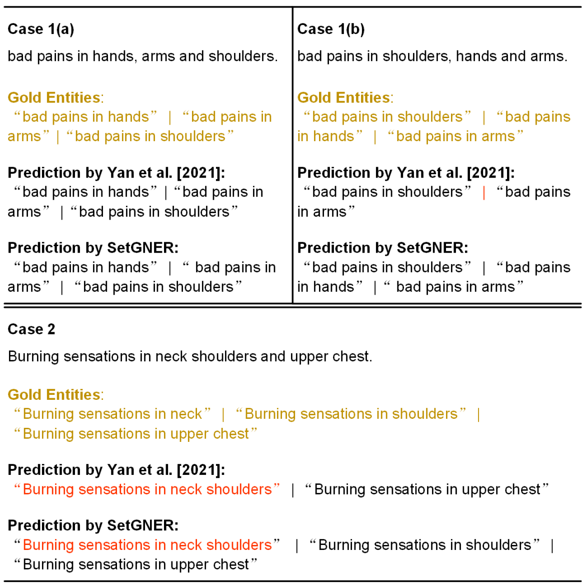
4.6.1 Recognizing Overlapped Entities

Since the sequential order assumed by seq2seq NER models is most untenable when faced with overlapped entities, we compare SetGNER with a seq2seq baseline model in terms of the ability to recognize overlapped entities. The experiment is conducted on the test sets of two discontinuous NER datasets and two nested NER datasets, where the overlaps between entities are common. We choose (Yan et al., 2021) as the baseline, since its overall performance on the four datasets is the best

Dataset	Scope	Yan et al. (2021)	SetGNER
CADEC	All	71.21	71.78 (+0.57)
	Overlapped	57.38	60.51 (+3.13)
ShARe13	All	77.42	77.82 (+0.40)
	Overlapped	59.90	62.16 (+2.26)
ACE04	All	86.41	86.69 (+0.28)
	Overlapped	84.57	85.16 (+0.59)
ACE05	All	86.38	86.40 (+0.02)
	Overlapped	84.31	85.08 (+0.77)

Table 5: Comparing SetGNER with baseline model (Yan et al., 2021) in terms of the ability to recognize overlapped entities (measured in Recall).

among SOTA seq2seq NER models. The results are shown in Table 5. We can see that SetGNER greatly boost the Recall of overlapped entities and the increase is several times higher than the overall increase of Recall on each dataset. This means Entity Set Generation is superior to seq2seq on the recognition of overlapped entities.


Case 1(a)
bad pains in hands, arms and shoulders.

Gold Entities:
"bad pains in hands" | "bad pains in arms" | "bad pains in shoulders"

Prediction by Yan et al. [2021]:
"bad pains in hands" | "bad pains in arms" | "bad pains in shoulders"

Prediction by SetGNER:
"bad pains in hands" | " bad pains in arms" | "bad pains in shoulders"

Case 1(b)
bad pains in shoulders, hands and arms.

Gold Entities:
"bad pains in shoulders" | "bad pains in hands" | "bad pains in arms"

Prediction by Yan et al. [2021]:
"bad pains in shoulders" | "bad pains in arms"

Prediction by SetGNER:
"bad pains in shoulders" | "bad pains in hands" | " bad pains in arms"

Case 2
Burning sensations in neck shoulders and upper chest.

Gold Entities:
"Burning sensations in neck" | "Burning sensations in shoulders" | "Burning sensations in upper chest"

Prediction by Yan et al. [2021]:
"Burning sensations in neck shoulders" | "Burning sensations in upper chest"

Prediction by SetGNER:
"Burning sensations in neck shoulders" | "Burning sensations in shoulders" | "Burning sensations in upper chest"

Figure 4: Cases from the CADEC dataset.

4.6.2 Removing Entity Order Confounder and Error Propagation

We conduct case study on the CADEC dataset to verify that SetGNER (w/o Inverse Generation Training) can overcome the Entity Order Confounder and the Error Propagation problem that seq2seq NER models suffer from. Figure 4 illustrates two cases from the dataset. In the first case, both SetGNER and the seq2seq baseline model (Yan et al., 2021) can correctly generate all the entities "bad pains in hands", "bad pains in arms" and "bad pains in shoulders". However, when we shuffle the order of "hands", "arms" and "shoulders" in the sentence, the seq2seq NER model fails to generate the entity "bad pains in hands". This means the seq2seq NER model is biased towards the original entity order that occurs more frequently in the training data. And SetGNER can get rid of this incorrect bias. In the second case, the seq2seq baseline model first generates the wrong mention "Burning sensations in neck shoulders", which consequently disturbs the generation of "Burning sensations in shoulders". In contrast, such a phenomenon of error propagation does not occur in our model.

Dataset	Right-boundary Acc.		Right-boundary Rec.	
	SetGNER	+ IGT	SetGNER	+ IGT
CADEC	72.74	74.86	72.93	73.34
ShARe13	81.99	83.45	77.67	78.22
ACE04	91.16	91.88	90.71	91.79
ACE05	89.67	90.68	91.18	91.58

Table 6: Comparing original SetGNER with SetGNER trained with Inverse Generation task (+ IGT) in terms of right-boundary accuracy and right-boundary recall.

4.6.3 Capturing Right-boundaries

We hypothesize that Inverse Generation Training is effective in teaching our encoder the right-to-left semantic structure. To verify this, we compare the ability of SetGNER to caputure right-boundaries before and after Inverse Generation Training. Concretely, right-boundary accuracy (# correct boundaries / # correct left-boundaries) and right-boundary recall (# correct right-boundaries / # golden right-boundaries) are measured on four benchmark datasets. As shown in Table 6, after Inverse Generation Training, the right-boundary accuracy of SetGNER increases by +2.12, +1.46, +0.72, + 1.01 and the right-boundary recall of SetGNER increases by +0.41, +0.55, +1.08 + 0.40 on the four datasets respectively. This demonstrates the effectiveness of Inverse Generation Training.

Methods	# sentences per second		
	CoNLL03	ACE04	ShARe13
Yan et al. (2021)	48 (1×)	23 (1×)	33 (1×)
Tan et al. (2021)	162 (3.4×)	74 (3.2×)	-
SetGNER	149 (3.1×)	65 (2.8×)	122 (3.7×)

Table 7: Efficiency comparison with SOTA seq2seq / seq2set NER models. Using a Nvidia RTX 3090 GPU.

4.7 Inference Efficiency

We compare the inference speed of SetGNER with the SOTA seq2seq NER model (Yan et al., 2021) and the SOTA seq2set NER model (Tan et al., 2021) on three datasets. For a fair comparison, we fix the maximum number of source tokens in each batch as 800 and ensure that the sentence sampling order in each run is the same. As shown in Table 7, SetGNER is about 3 times faster than the seq2seq NER model, thanks to the distributed nature of our design. However, SetGNER is still slower than the seq2set NER model based on non-autoregression.

5 Related Work

Seq2seq NER Models Straková et al. (2019) propose to linearize BILOU labels (Ratinov and Roth, 2009) of source tokens into a target sequence. Their linearization of BILOU labels follows a heuristic rule, which may introduce incorrect model bias.

Athiwaratkun et al. (2020) propose an augmented natural language output format for flat NER, where the type tags of words are placed along with the words to form a sentence-alike target sequence. Lu et al. (2022) represent different information structures with a structured extraction language and solve general information extraction tasks with a unified text-to-structure generation framework. Zhang et al. (2022) point out two kinds of incorrect bias (pre-context confounder, entity-order confounder) in the seq2seq NER models and propose two data augmentation methods to address them. However, their model is still trained to generate entities sequentially after all.

There also exist pointer-based target sequences. Fei et al. (2021) train a LSTM from scratch to generate the target sequence and devise a novel memory-augmented pointer mechanism to encourage interactions between the current pointer and the prior recognized entity mentions. Instead, Yan et al. (2021) combine pre-trained BART with a delicately-designed copying mechanism and achieve promising performance on a wide range of

NER benchmarks. Our work inherits the copying mechanism proposed in this work.

The seq2set NER model Tan et al. (2021) observe that nested NER is essentially an unordered recognition task and propose to predict the set of entity *spans* in one pass via a non-autoregressive model. In contrast, our model treats each entity as a sequence rather than a span, and is able to handling discontinuous entity mentions.

6 Conclusion

We observe that existing generative NER models suffer from the entity order confounder and faces the problems of error propagation and slow inference speed. To address this, a novel Entity Set Generation framework for general NER is proposed in this paper. We also propose to train our model with an auxiliary Inverse Generation task that helps the encoder learn the right-to-left semantic structure. Experiments on five datasets prove the effectiveness of our methods.

Limitations

Since both the encoding module and decoding module of SetGNER work in the word-level, SetGNER requires the input sentence to be tokenized into words beforehand. However, for many language, e.g. Chinese and Japanese, how to conduct word segmentation and whether it is necessary to do so are still open questions. This limits the usage of SetGNER. To adopt SetGNER to these language, further research is required.

Another limitation of SetGNER is that when generating the set of entity sequences, there is not interaction between target sequences starting with different head words. This may limit the model's performance when explicit information about other entities is helpful for the recognition of the target entity.

Last but not least, although SetGNER is about 3 times faster than the SOTA seq2seq NER model (Yan et al., 2021), it consumes more memory. The maximum memory occupation of SetGNER is about 2 times larger than that of (Yan et al., 2021), when the maximum number of source tokens in a batch is set as 800. This should be taken into account when deploying the model on machines with small graphical memory.

Acknowledgments

We thank the reviewers for their insightful comments and valuable suggestions. This study is partially supported by National Key R&D Program of China (2021ZD0113402), National Natural Science Foundations of China (62276082, U1813215 and 61876052), National Natural Science Foundation of Guangdong, China (2019A1515011158), Major Key Project of PCL (PCL2021A06), Strategic Emerging Industry Development Special Fund of Shenzhen (20200821174109001) and Pilot Project in 5G + Health Application of Ministry of Industry and Information Technology & National Health Commission (5G + Luohu Hospital Group: an Attempt to New Health Management Styles of Residents).

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 724–728.

Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–385, Online. Association for Computational Linguistics.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2020. An effective transition-based model for discontinuous NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5860–5870.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ace) program tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*.

Hao Fei, Donghong Ji, Bobo Li, Yijiang Liu, Yafeng Ren, and Fei Li. 2021. Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12785–12793.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of Biomedical Informatics*, 55:73–81.

David Kemmerer. 2015. *Cognitive Neuroscience of Language*. Psychology Press.

Mike Lewis, Yinhan Liu, and Goyal. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4814–4828.

Xiaoya Li, Jingrong Feng, and Yuxian Meng. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, and Zeqi Tan. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 2782–2794.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W. Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R. South, Danielle L. Mowery, Gareth J. Jones, Johannes Leveling, Liadh Kelly, Lorraine Goeuriot, David Martinez, and Guido Zuccon. 2013. Overview of the share/clef ehealth evaluation lab 2013. In *Proceedings of the 4th International Conference on Information Access Evaluation. Multilinguality, Multimodality, and Visualization - Volume 8138*, CLEF 2013, page 212–231, Berlin, Heidelberg. Springer-Verlag.

Zeqi Tan, Yongliang Shen, and Shuai Zhang. 2021. A sequence-to-set network for nested named entity recognition. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI-21*.

Buzhou Tang, Jianglu Hu, Xiaolong Wang, and Qingcai Chen. 2018. Recognizing continuous and discontinuous adverse drug reaction mentions from social media using lstm-crf. *Wireless Communications and Mobile Computing*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Christopher Walker, Stephanie Strassel, and Kazuaki Maeda. 2006. The automatic content extraction (ace) program tasks, data, and evaluation. In *Linguistic Data Consortium*, page 57, Philadelphia.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021a. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Yucheng Wang, Bowen Yu, Hongsong Zhu, and Tingwen Liu. 2021b. Discontinuous named entity recognition as maximal clique discovery. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 764–774.

Yongxiu Xu, Heyan Huang, Chong Feng, and Yue Hu. 2021. A supervised multi-head self-attention network for nested named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14185–14193, Online.

Hang Yan, Tao Gui, and Junqi Dai. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Shuai Zhang, Yongliang Shen, Zeqi Tan, Yiquan Wu, and Weiming Lu. 2022. De-bias for generative extraction in unified NER task. In *Proceedings of the*

60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 808–818, Dublin, Ireland. Association for Computational Linguistics.

A Profile of Datasets

Statistics	ACE04			ACE05		
	Train	Dev	Test	Train	Dev	Test
# Sentences	6200	745	812	7194	969	1047
Avg sent. length	22.5	23.0	23.0	19.2	18.9	17.2
# Entities	22204	2514	3035	24441	3200	2993
# Nested entities	10149	1092	1417	9389	1112	1118

Table 8: Statistics of nested NER datasets ACE04 and ACE05.

Statistics	CADEC	ShARe13
# Sentences	7597	18767
Avg sent. length	14.2	12.9
# Entities	6318	11161
# Overlapped entities	923	663
# Discontinuous entities	675	1090

Table 9: Statistics of discontinous NER datasets CADEC and ShARe13.

Statistics	CoNLL03		
	Train	Dev	Test
# Sentences	14041	3250	3453
Avg sent. length	13.7	13.5	13.6
# Entities	23326	5902	5613
PER	6532	1829	1597
LOC	7125	1832	1644
ORG	6271	1325	1654
MISC	3398	916	5613

Table 10: Statistics of flat NER dataset CoNLL03.

CADEC[1] (Karimi et al., 2015) is a discontinuous NER dataset with a corpus of adverse drug event. It originally contains 5 entity type, but only annotations of "ADE" entities are considered. Because only ADEs include discontinuous entities.

ShARe13[2] (Suominen et al., 2013) is a discontinuous NER dataset with a corpus of clinical notes and contains annotations of disorder mentions.

ACE04[3] **and ACE05**[4] (Doddington et al., 2004; Walker et al., 2006) are two nested NER dataset with corpuses of newswire, broadcast news and telephone conversations. Both of them contains 7 entity categories: “PER”, “ORG”, “LOC”, “GEP”, “VEH”, “WEA” and “FAC”.

CoNLL03[5] Tjong Kim Sang and De Meulder (2003) is a flat NER dataset with a news corpus and has annotated 4 types of entities as “PER”, “LOC”, “ORG” and “MISC”.

Statistics of the five datasets are listed in Tabel 8-10.

B Hyper-parameter Settings

We manually tune the hyper-parameters for each dataset. Specifically, we trial different values of each hyper-parameter within a bound and the hyper-parameter value that results in the best performance (measured in F1-score) on the development set are chosen. The search bound of each hyper-parameter and the final hyper-parameter configuration are shown in Table 11.

C Sensitivity Analysis

The position interval for adaptive position embedding, τ, and the beam size of adaptive beam search, β, are two important hyper-parameters of SetGNER. To analyze their influence on the performance of SetGNER, we experiment with different values of them and record the corresponding F1 scores on four datasets (CADEC, ShARe13, ACE04 and ACE05). As shown in Figure 5, when $\tau = 0$ (the adaptive position embedding degrades to vanilla position embedding), the model cannot work properly, demonstrating the necessity of adaptive position embedding for SetGNER. When $\tau \geq 1$, the change of τ slightly affects the performance of SetGNER by a margin of around $0.2 \sim 0.4$ F1 on the four datasets. Figure 6 shows the F1 scores of SetGNER with different beam sizes. We can see that the performance of SetGNER reaches the peak when the beam size is around $4 \sim 6$, and does not further improve when the beam size grows bigger.

D Compared Discriminative NER Models

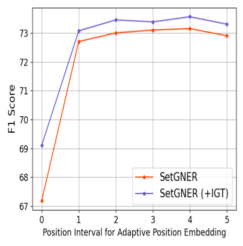
D.1 Models for Discontinuous NER

Tang et al. (2018) use LSTM-CRF to recognize continuous and discontinuous adverse drug reaction mentions. (Dai et al., 2020) is a transition-based

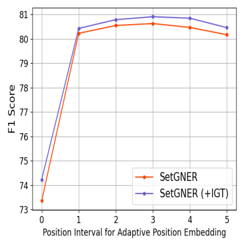
[1]https://data.csiro.au/collection/10948v003
[2]https://physionet.org/content/shareclefehealth2014task2
[3]https://catalog.ldc.upenn.edu/LDC2005T09
[4]https://catalog.ldc.upenn.edu/ LDC2006T06
[5]https://www.clips.uantwerpen.be/conll2003/ner/

Hyper-parameters	Bound	CADEC	ShARe13	ACE04	ACE05	CoNLL03
Epoch	[30, 60]	55	50	55	50	50
Warmup ratio	[0.001, 0.2]	0.1	0.1	0.01	0.01	0.01
$\text{lr}_{\text{encode}}$	[5e-6, 2e-5]	7.5e-6	7.5e-6	1.2e-5	1e-5	7.5e-6
$\text{lr}_{\text{decode}}$	[5e-6, 2e-5]	1e-5	1e-5	1.6e-5	1.3e-5	1e-5
Max tokens per batch	[150, 400]	280	280	190	220	400
Encoder dropout	[0.001, 0.05]	0.03	0.03	0.03	0.03	0.03
Decoder dropout	[0.001, 0.05]	0.001	0.001	0.02	0.01	0.05
Position interval τ	[0, 5]	4	3	2	4	4
Beam size β	[1, 8]	4	4	4	4	1

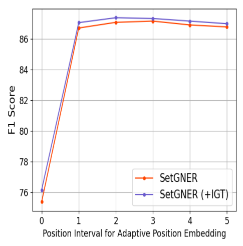
Table 11: The hyper-parameter settings in our experiments.



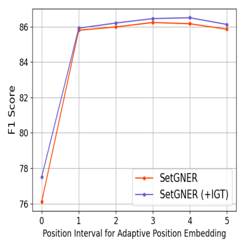
(a) CADEC



(b) ShARe13

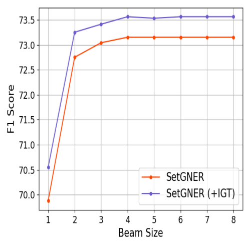


(c) ACE04

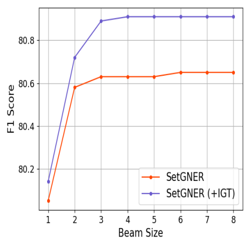


(d) ACE05

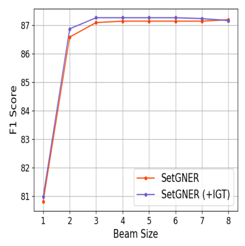
Figure 5: F1 scores under different position interval τ.



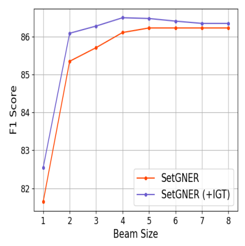
(a) CADEC



(b) ShARe13



(c) ACE04



(d) ACE05

Figure 6: F1 scores under different beam sizes β.

method that utilizes shift-reduce parsers to identify discontinuous entities. Wang et al. (2021b) solve discontinuous NER via the maximal clique discovery algorithm based on graph theory.

D.2 Models for Nested NER

Yu et al. (2020) formulate NER as the dependency parsing task and solve it with TreeCRF. (Li et al., 2020) is a method based on machine reading comprehension. Xu et al. (2021) treat named entity recognition as multi-class classification of spans and solve it with a multi-head self-attention mechanism. (Shen et al., 2021) is a two-stage entity identifier, which first generates candidate spans and then labels the boundary-adjusted span proposals with the corresponding categories.

D.3 Models for Flat NER

Akbik et al. (2019) dynamically aggregate contextualized embeddings of each encountered string and use a pooling operation to distill a global word representation from all contextualized instances. Wang et al. (2021a) use the input sentence as a query to retrieve external contexts with a search engine and concatenate the sentence with its external contexts.